Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

PREDICTIVE-CONSERVATIVE SYNCHRONIZATION FOR COMMERCIAL SIMULATION PACKAGE INTEROPERABILITY

Yuanxi Liang
Stephen John Turner

School of Computer Engineering
Nanyang Technological University
Singapore 639798

Boon Ping Gan

D-SIMLAB Technologies Pte Ltd
9 Jurong Town Hall Road #03-45 iHUB
Singapore 609431

ABSTRACT

Distributed simulation is desired in many industries to support analysis and decision making for complex and integrated problems. Interoperating commercial simulation packages using the High Level Architecture (HLA) is a technique to fulfill this demand. Time management is an important factor in determining the execution performance when using this technique. In some simulation models, conservative synchronization does not provide good efficiency because of the high cost of requesting a time advance with every event. In this paper, we describe a user-friendly framework for interoperating AutoMod models by adopting the HLA standard with a new time management mechanism. A "departure port" mechanism identifies the characteristics of the message outgoing process and provides a dynamic safe request time to invoke the time advancing service. We conducted experiments using close-to-reality models which show that the execution time can be reduced by over 50%.

1 INTRODUCTION

In the last decade, simulation has become a vital tool in many industries for business analysis and forecasting in response to the fast changing business environment. Many commercial off-the-shelf simulation packages (CSPs) are available in the market for simulation and modeling, each with their own particular strengths. To investigate large-scale complex problems and support decision-making, interoperating different CSPs using the High Level Architecture (HLA) has become increasingly important. This technique allows a complex simulation system to be formed by reusing different simulation components and applications (IEEE 1516. 2000).

In the semiconductor manufacturing industry, simulation is widely used in production line analysis and planning. As the wafer fab operations have become increasingly complex, a much larger and complex simulation model is needed. AutoMod which is developed by Brooks Software Corporation (Brooks Atuomation 2003) is widely used in the modeling of manufacturing and automated material handling systems (AMHS) (Rohrer 2000), while AutoSched AP is used to model the manufacturing process (MP). Interoperating AutoSched AP models using the HLA standard was carried out by Gan et al to model the MP (2005a). However, with the trend towards 300 mm wafer fabs, it becomes critical to consider the AMHS in the simulation model as 300 mm fabs rely on AMHS for operational efficiency (Gan et al. 2006). Interoperating AutoMod and AutoSched AP using the HLA standard becomes an ideal approach.

Time management is an important factor in determining the execution performance of the distributed simulation. However, the current approach of using conservative synchronization does not provide good efficiency in certain simulation models. This is due to the high cost of requesting a time advance with every event. This paper describes our work in interoperating AutoMod using the HLA and using a new time management approach to improve the execution efficiency.

This paper is organized as follows: Section 2 gives an overview of related work in terms of current achievements and existing problems in interoperating AutoMod. Section 3 describes the new time management mechanism which is called predictive-conservative synchronization, the departure port concept, and its implementation. In section 4, the AutoMod HLA Interface (AMHI) and the AutoMod HLA Development Kit (AHDK) are briefly introduced. Following that, the experiments and results are discussed in Section 5. Finally, Section 6 gives conclusions and future work.

2 RELATED WORK

2.1 CSP Interoperability

In order to integrate different CSPs based on the HLA, a generic architecture (Figure 1) has been defined by the CSP Interoperability Product Development Group (CSPI-

PDG) which provides a standard for interoperation through a middleware approach (CSPI-PDG 2008). There are five requirements for using this architecture (Gan et al. 2006):

- The ability to initialize the distributed simulation prior to simulation execution
- The ability to suspend the simulation execution
- Access to the time of the next event to be simulated
- The ability to introduce new events/entities from an external source into the event list
- Access to information of simulation objects/entities that are shared among federates

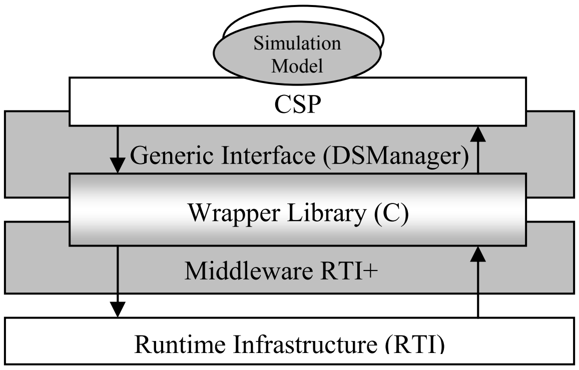


Figure 1: A generic architecture for integration of CSPs with the HLA

With a generic interface called DSManager, the generic architecture has been applied to interoperate AutoMod with AutoSched AP in order to accurately simulate a 300 mm wafer fab in the semiconductor manufacturing industry. However, the time management of conservative synchronization used in the experiment led to a low execution efficiency as discussed in Gan et al. (2006).

2.2 Time Management in the HLA

In this paper, we focus on event-driven simulations where the federate's logical time advances to the time stamp of each event as it is processed. The Runtime Infrastructure (RTI), which is software that allows a federation to execute together (Dahmann et al. 1998), offers time advancing services Next Event Request (NER) and Next Event Request Available (NERA) for event-driven federates. Both services take a request time from the simulation federate and will grant the simulation federate a time that is less than or equal to the request time. After receiving a grant time T from the RTI in response to an NER service call, the federate will not receive any message with time stamps less than or equal to T. However, for an NERA service call, the grant time returned does not guarantee that the federate will not receive any message with time stamp exactly equal to T.

Moreover, lookahead is a factor that needs to be considered in choosing between NER and NERA. This is a parameter associated with each federate and states that the federate will not send out any message with timestamp less than the federate's current logical time plus the lookahead value. Lookahead is generally a fixed value and plays a role in reducing the number of NER/NERAs in a federate. NERA is generally used when the federate's lookahead is zero (Fujimoto 1998).

In general, when conservative synchronization is used for time management, a request to advance time is issued for every event in the future event list. This approach may cause a large synchronization overhead and slow down the execution of the simulation. This paper describes an approach that may be used to reduce this overhead.

2.3 Interoperating AutoMod

The simulation events are classified into two categories, which are external events and internal events (Gan et al. 2006). The external events refer to the events that are associated with the action of sending messages to other federates, while the internal events do not send out any message to other federates. The AutoMod model used in the experiment in Gan et al. (2006) has many more internal events than external events. Using conservative synchronization results in low execution efficiency because of the large number of NERs from the AutoMod model. Ideally, it is only necessary to issue a request to advance time for an external event. NER is a promise that a federate will not send any message with time stamp less than the value of requested time plus lookahead if the requested time is granted. Hence, NERs for internal events are not necessary as the internal events do not affect the calculation of the grant time in the RTI.

Previous work also includes interoperating AutoSched AP models by using an optimized time management algorithm which is based on a particular manufacturing scenario (Gan et al. 2005b). The basic idea of the algorithm is to issue an NER according to the time stamp of the next external event. The times of the potential external events are placed in a queue. When the queue is not empty, the NER will use the time stamp of the first potential external event as the request time. This mechanism reduces the number of NERs to the RTI, and it gains better performance than the usual conservative synchronization. However, this approach is specific to a particular model that only has one kind of external event, and needs to be extended for more generic models (Gan et al. 2005a).

The previous work inspires the idea in this paper which is to reduce the number of NER/NERAs. Many models, especially models of manufacturing processes and automated material handling systems, have the characteristic that we can predict the time of the next external event based on both dynamic and static information provided by the model itself.

3 PREDICTIVE-CONSERVATIVE SYNCHRONIZATION

3.1 Overview

In order to improve the execution efficiency of AutoMod using the HLA, the new time management mechanism predictive-conservative synchronization is proposed. This predicts the time of external events based on both dynamic and static information provided by the model itself, and invokes time advancing services by making use of the predicted values. When the federate is ready to advance its logical time, the request time to the RTI will be calculated based on information of all current predictions, and the request time need not be the time of the next event in the future event list. Predictive-conservative synchronization aims to invoke the time advancing service with the logical time of the next external event, and eliminate all non-critical requests to advance time for internal events.

3.2 Departure Port

The Departure Port is an abstract concept which describes the entity outgoing process and behaviors of a model. The entity is represented by a message and is the basis of information exchange between models (Wang et al. 2004). Each departure port is defined by two points, the entrance point and the exit point. The entrance point is the condition (internal event) that will trigger an external event, while the exit point is the condition (external event) where entities are sent out. There is an attribute associated with each departure port called the "fixed minimum processing time" (FMPT), which is the minimum period of logical time between the entrance point and the exit point.

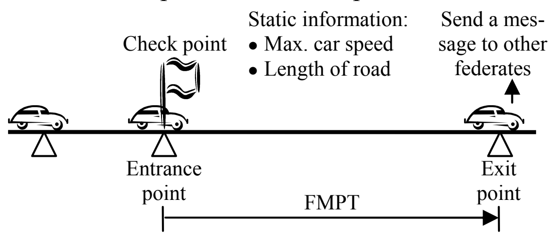


Figure 2: Illustration of departure port concept

For example (Figure 2), a model simulates a car moving on a road. When the car reaches the end of the road, this federate will send a message to other federates. In this case, the external event is the car reaching the end of the road, and the internal event which will trigger the external event is the car reaching the check point. The departure port can be defined as shown in Figure 2, and the FMPT can be calculated based on the maximum car speed and the length between the check point and the end of the road. On the other hand, the dynamic information in this example will be discussed in section 3.3.1.

3.2.1 Characteristics and Restrictions

Each departure port can be associated with one or more than one type of outgoing entities.

There are three restrictions when identifying a departure port in a model. First of all, each departure port forms a First-In-First-Out (FIFO) external event queue. The expected happening times of external events are put into the queue in time stamp order. The expected time of the external event at the top of the queue should have the smallest value among all the expected times in the queue.

Moreover, each departure port can only have one entrance point, and one exit point. Once the outgoing entity is inside the departure port, it is confirmed that this entity will be sent out at the exit point. The modeler needs to choose the entrance point carefully to satisfy this restriction.

Lastly, the fixed minimum processing time of a departure port is a fixed value which is the minimum period of logical time between the entrance point and the exit point under all possible situations. However, during the simulation, the actual processing time from the entrance point to the exit point can be larger than this fixed minimum processing time.

3.2.2 The Departure Port Design Methodology

It is necessary to design the departure port in a model before using predictive-conservative synchronization. A basic criterion is that all types of outgoing entity should be associated with at least one departure port, so that all the external events of outgoing processes are monitored during the simulation.

In general, it is first necessary to identify the condition and place where the entity will be sent out. Then the condition (internal event) that will trigger this external event is determined. Finally, the fixed minimum processing time of this departure port is calculated based on the behaviors of the model between the entrance point and the exit point. In summary, we can follow a three-step approach, which includes

1. Define the exit point
2. Define the entrance point
3. Calculate the fixed minimum processing time

3.3 Three Phases to Invoke NER/NERAs

The predictive-conservative approach consists of three phases, the predicting phase, the time advancing phase and the sending phase.

3.3.1 Predicting Phase

Each departure port is associated with a "potential event queue" (PEQ), which is a basic FIFO queue data structure containing the expected times of the external events. When the internal event at the entrance point occurs, the expected time (ET), at which we expect the external event to happen, will be pushed into the associated potential event queue. The model designer needs to ensure that the expected time is not greater than the actual happening time of the external event when they choose the prediction method, otherwise, an error may occur.

With the same example as 3.2 (Figure 3), when a car passes the check point, the expected time of the car reaching the end of the road will be made based on both dynamic and static information and pushed into the FIFO queue. In this case, we assume the car speed is constant and this information is dynamic because different cars have different speeds.

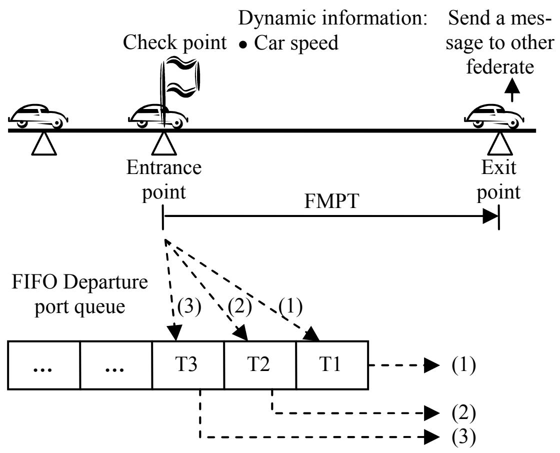
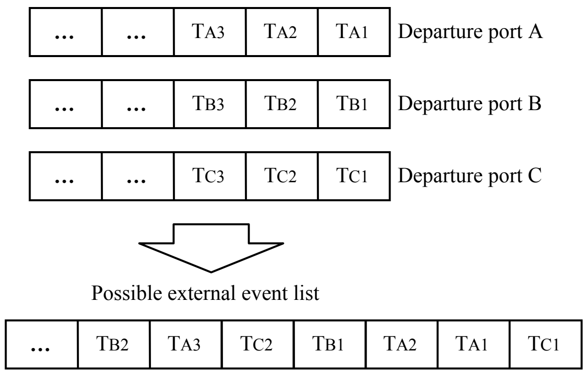


Figure 3: Illustration of predicting and sending phases

3.3.2 Time Advancing Phase

With all the predictions, we can form a possible external event list (Figure 4) which provides the expected time of all the potential external events in a model. The federate can invoke the time advancing service with the smallest expected time in the list. The algorithm is able to detect whether the prediction is accurate by comparing the time of the next event and the request time calculated from the predictions. In the case that the federate has invoked the time advancing request with an inaccurate expected time, the algorithm is able to switch back to normal conservative synchronization to ensure the correctness of the synchronization.

Figure 5 shows the pseudo code for the time advancing phase. The granted time (GT) is the value given by the RTI, which indicates that the federate will not receive any message with time stamp less than this value from other federates. If the time of the next event (NET) on the future event list is less than the time granted by the RTI, the next event is safe to be processed and there is no need to invoke the time advancing service.



Note: Txx is the value of the expected time of an external event

Figure 4: A possible external event list

```
1.  While not end of simulation
2.    NET = time of next event in the future
        event list
3.    If (NET >= GT)* Then
4.       RT = call getRequestTime()
5.       If (NET > RT) Then
6.         RT = NET
7.       End If
8.       GT = invoke time advancing service
         (NER/NERA) with RT to RTI
9.    End If
10. End of while loop

*: When the application uses NER service,
line 3 should be "If (NET > GT) Then".
When the application uses NERA service,
line 3 should be "If (NET >= GT) Then".
```

Figure 5: Time advancing phase

Otherwise, we need to invoke the time advancing service of the RTI. The function "getRequestTime()" calculates the request time based on all the prediction times. The request time (RT) is the expected time of the next external event which will happen in the future (Figure 5, line 4). When the prediction time made in the predicting phase is 100% accurate and all the departure ports are not empty, the RT should be greater than or equal to the NET. In this case, the federate will invoke an NER/NERA with the value of RT. However, if the prediction time is inaccurate or some of the departure ports are empty, the NET may be greater than the RT. In this case, we will use the NET value as the request time to invoke an NER/NERA (Figure 5, line 6). This function "getRequestTime()" is called before processing every event on the future event list.

Figure 6 shows how the request time is decided inside the getRequestTime() function. The basic approach is to find out the expected time of the first external event from each departure port, and then look for the first external event of the whole model among these expected time values.

```
1.  Function getRequestTime()
2.    RT = 0
3.    If (NumPEQ > 0) Then
4.      For i = 1 To NumPEQ
5.        If (PEQ[i].IsEmpty()) Then
6.          PRT[i] = currentTime + FMPT[i]
7.        Else
8.          PRT[i] = PEQ[i].peek()
9.        End If
10.     Next i
11.     RT = minimum value among PRT[] array
12.   End If
13.   Return RT
14. End Function
```

Figure 6: Function getRequestTime()

The possible request time of each potential event queue (PRT[i]) is the minimum value of the expected time in each potential event queue, in the case that the queue is not empty (Figure 6, line 8). The function peek() only returns the value of the first event on the queue, but does not remove it from the queue. It is used here because this algorithm needs to keep the prediction until the associated external event happens. If the queue is empty, the PRT[i] will be the sum of the current logical time of the federate and the fixed minimum processing time (FMPT[i]) of the associated departure port which is defined in the model designing stage (Figure 6, line 6). Finally, the request time is the minimum value among all those possible request time values.

Predictive-conservative synchronization is different from previous work on conservative synchronization like the “pre-sending” concept (Nicol 1993). The “pre-sending” concept assumes the completion time of a process is known at the beginning of the process, but it must be 100% accurate. In our approach, the completion time is predicted based on both static and dynamic information and need not be 100% accurate. Moreover, compared to a general lookahead, the FMPT defines a processing time for each departure port and may have a value of zero. The RT is calculated based on all the dynamic information stored at the departure ports and only makes use of the FMPT when a specific departure port is empty. Even when all departure ports have zero FMPT, this algorithm still serves to improve the execution efficiency using the dynamic prediction of external events. Thus, when we use this approach, the external lookahead of the federate is usually set to zero, and the federate invokes the NERA, because the role of lookahead has been replaced by both the static and dynamic information gathered in departure ports.

3.3.3 Sending Phase

In the sending phase, the prediction time is removed when the external event happens. When this happens, the associated expected time becomes outdated, and we have to remove it from the potential event queue.

With the same example in Figure 3, when the car reaches the end of the road, the associated prediction will be popped out from the queue in the sending phase.

3.4 Implementation

Predictive-conservation synchronization can be applied to any CSP which complies with the five requirements of the generic architecture discussed in section 2. The core of the implementation is to establish a queue data structure for every departure port. In principle, the potential event queue can be implemented in the CSP, the CSP interface which is a bridge between the CSP and the generic interface, and the generic interface itself which is the DSManager in our paper (Figure 7).

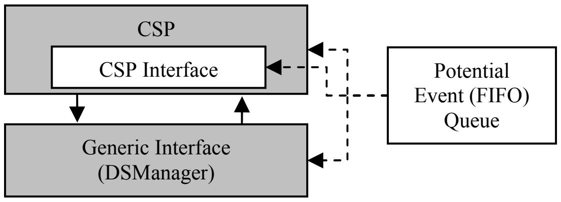


Figure 7: Possible implementations

The first option can make use of the existing queue component available in the CSP. However, it may not be flexible to manipulate the queues. For the second option, upon running the simulation, a number of queues can be established inside the CSP interface depending on the number of departure ports in the model itself. This is preferred because of the reusability of the data structure. The third option reduces the work in designing the CSP interface, but the queue data structure may be hard to implement in the generic interface. The enqueue operation in the predicting phase and the dequeue operation in the sending phase are both triggered within the CSP.

In practice, the second option is preferred because the CSP interface is dependent on the CSP and it is more convenient to obtain information from the model, while the generic interface is independent of the CSP.

4 INTEROPERATING AUTOMOD USING THE HLA

AutoMod satisfies the five interoperation requirements as discussed in Gan et al. (2006). The previous version of the interface between AutoMod and the DSManager library which is coded in Visual Basic 6.0 has been further developed by adopting the predictive-conservative synchronization. The AutoMod HLA Interface (AMHI) is a CSP interface including a basic queue data structure which is used to store the expected happening time of external events for every departure port, and it has a user friendly GUI to let the user specify necessary settings.

Moreover, the AutoMod HLA Development Kit (AHDK) was developed with the purpose of helping designers in the model development stage. The code for synchronization is automatically generated into the model. An advantage is that all the time management implementation details are hidden from the model designer, so it minimizes the effort of understanding the synchronization mechanism.

5 EXPERIMENTS

5.1 Overview

We might expect that a longer processing time of a departure port would give us a better performance in terms of reduction in execution time and the number of NER/NERAs. We designed two experiments based on AutoMod to evaluate how much improvement can be achieved by using predictive-conservative synchronization, and how the length of the processing time and the prediction accuracy impact on the performance.

The experiments were conducted by using two computers to run AutoMod models, and one computer to run the RTI server process. All these computers are interconnected by a 100 Mbps Ethernet.

5.2 Experiment I

5.2.1 Experiment Models

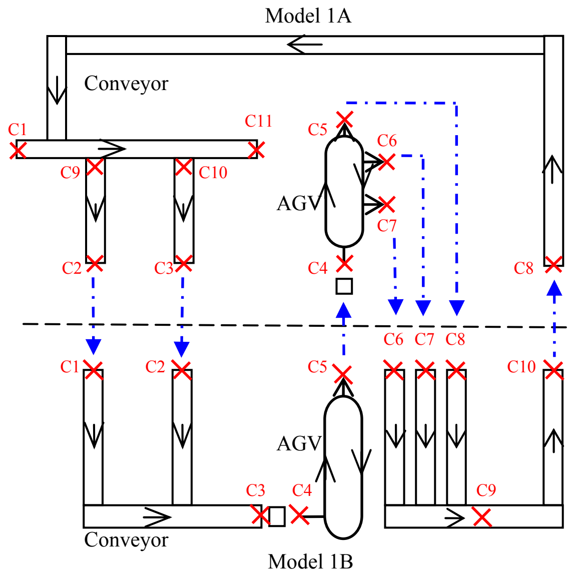


Figure 8: The Experiment I models

Figure 8 shows the two distributed models in Experiment I. In each model, there is a conveyor system and an automated guided vehicle (AGV) system. The cross symbol are the control and reference points which indicate the location in the system, and each control point has a unique name in AutoMod. The handling system starts at C1 in Model 1 conveyor system where lots are generated with a given distribution at C1. The lots will travel between these two models. After the lots come back to Model 1 at C8, the majority of the lots will travel along the additional track and finish processing at C11, but some of the lots will be transferred to C2 or C3 and the whole process repeats again. These two models form a typical handling system in the manufacturing scenario, as the conveyor system and the AGV system are commonly used in many industries.

5.2.2 Departure Port Design

Table 1: Departure port design version I

	Departure Port ID	Entrance	Exit
Model 1A	1	C9	C2
	2	C10	C3
	3	C4	C5
	4	C4	C6
	5	C4	C7
Model 1B	1	C4	C5
	2	C9	C10

Table 2: Departure port design version II

	Departure Port ID	Entrance	Exit
Model 1A	1	C1	C2
	2	C1	C3
	3	C4	C5
	4	C4	C6
	5	C4	C7
	6	C8	C2
	7	C8	C3
Model 1B	1	C4	C5
	2	C6	C10
	3	C7	C10
	4	C8	C10

We designed two sets of departure ports. Table 1 shows the first version of the design which has five and two departure ports in Model 1A and Model 1B respectively. In the second version of the design (Table 2), we have seven and four departure ports in Model 1A and Model 1B respectively. The difference between these two design approaches is the length of the departure port, as well as the length of the predicted processing time. The second approach has larger prediction time values in some of the departure ports than those in the first approach. Although some departure ports have the same entrance point, the model provides necessary information to assign the entity into different departure ports when it arrives at the entrance point.

The prediction of the processing time in the departure port is made based on the length of the track and the speed of the conveyor and the vehicle. However, this prediction may not be accurate due to the delay on a connection between two conveyor sections (Brooks Automation 2003) referred to as the junction effect. In the AGV system, the acceleration and deceleration time is hard to calculate so it also lowers the prediction accuracy. Moreover, the turning effect when the vehicle makes a turn on the track must also be considered. To eliminate the turning effect, we can add the turning time in the prediction.

To study the effect of the inaccuracy of the prediction, we designed two prediction methods. The first prediction method did not eliminate the turning effect of the vehicle and has a low accuracy level, while the second prediction method has a higher prediction accuracy.

5.2.3 Experiment Results

Each experiment was executed for a simulation time of one day, and the results are the average values over three runs. The variation is small because the experiments were conducted on dedicated computers. As results of Model 1A and Model 1B are similar, only Model 1A's performance measurements are shown in this paper.

We measured the execution time, the total number of NERAs of each federate, and the number of outdated predictions which is the number of NERAs where the time of the next event is greater than the computed request time. The percentage of the number of outdated predictions to the total number of time requests is a way to measure the accuracy level of the predictions. If all the predictions are accurate in a federate and every departure port always has an event time in the potential event queue, this percentage will be zero. The higher the value of the percentage of outdated predictions, the lower prediction accuracy level we have.

Table 3: Percentage of NERAs reduced in Model 1A

	% of NERA Reduced	
	Low Prediction Accuracy	High Prediction Accuracy
Version I	80.17%	89.85%
Version II	77.22%	86.63%

Table 4: Percentage of outdated predictions in Model 1A

	% of Outdated Predictions	
	Low Prediction Accuracy	High Prediction Accuracy
Version I	58.37%	22.29%
Version II	68.81%	49.90%

Table 5: Percentage of execution time reduced

	% of Execution Time Reduced	
	Low Prediction Accuracy	High Prediction Accuracy
Version I	52.22%	60.89%
Version II	51.86%	59.11%

From Table 3 and Table 5, we see that by adopting predictive-conservative synchronization, the total number of NERAs in Model 1 and the execution time are reduced significantly. Based on the same departure port design, eliminating the turning effect of the vehicle, which gives a better prediction accuracy level, results in a better performance compared to the low prediction accuracy with the turning effect. However, Table 4 shows that version II of the departure port design has worse prediction accuracy than version I whether the turning effect is eliminated or not. The prediction accuracy level drops if we make the processing time of the departure port longer. In this experiment, this drop in accuracy may be due to factors like the junction effect in a conveyor system and the acceleration and deceleration effect in the AGV system.

In general, a higher accuracy in prediction gives a better performance in terms of the reduction in the number of NERAs and the execution time. However, Experiment I does not give an answer to the relationship between the length of the processing time and the performance when the prediction accuracy is unchanged.

5.3 Experiment II

5.3.1 Experiment Models and Departure Port Design

In Experiment I, the prediction accuracy level is dependent on the length of departure port. To investigate the relationship between the length of processing time and the performance based on the same prediction accuracy level, we built a simple distributed system where the prediction accuracy level is independent of the length of departure port. In this system, we only used a conveyor system with separated sections to eliminate the junction effect and to achieve 100% prediction accuracy. Figure 9 shows the illustrations of these two models, Model 2A and Model 2B, and their associated control points. We added in more control points with the purpose of simulating extra internal events between each entrance point and the exit point. Transfer of entities between models is shown by the dotted line. Based on this new system, we also designed two versions of departure ports (Table 6 and Table 7). The first version has a small processing time for departure ports, while the second version has a large processing time.

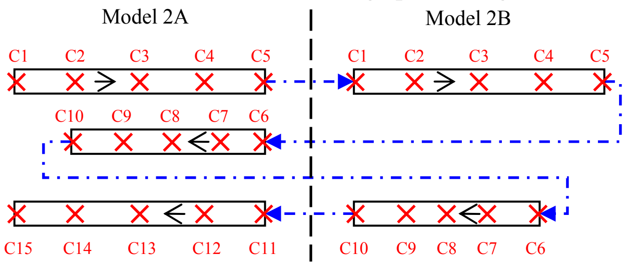


Figure 9: The Experiment II models

Table 6: Departure port design version I

	Departure Port ID	Entrance	Exit
Model 2A	1	C4	C5
	2	C9	C10
Model 2B	1	C4	C5
	2	C9	C10

Table 7: Departure port design version II

	Departure Port ID	Entrance	Exit
Model 2A	1	C1	C5
	2	C6	C10
Model 2B	1	C1	C5
	2	C6	C10

5.3.2 Experiment Results

Each experiment was executed for a simulation time of three days, and the results are the average values over three runs. As Model 2A and Model 2B have similar results, only Model 2A's performance measurements are shown in this paper.

In Experiment II, first, we ran the model using 100% accurate predictions, and then compared the performance results using version I of the departure port design with the results using version II of the design.

The second step was to purposely introduce some offsets in the prediction of both models so as to make the predictions inaccurate. Therefore, we decreased the expected time value of the external event by a few seconds in both models, and made it such that the prediction accuracy level in both models were roughly the same.

Table 9 shows that the prediction accuracy level in both departure port design version I and version II were the same. At the same prediction accuracy level, from Table 8 and 10, we see that using the version II departure port design, which has a longer length of departure port than version I of the design, gave better performance in terms of percentage of NERAs reduced and percentage of execution time reduced. A design with large processing time of departure port is preferred when the prediction accuracy level does not vary with different designs.

In summary, first, when the prediction accuracy level does not change with different design of departure port, the departure port design with a large predicted processing time will give better performance than a design with a small predicted processing time. Secondly, in the same design of departure port, making a prediction with a high accuracy will give better improvement than having a low accuracy prediction.

In practical situations, the departure port design with a larger processing time may have a lower prediction accuracy. Hence, we should make a departure port with a larger value of processing time, but we should also make the prediction as accurate as possible. If a departure port with a larger value of processing time reduces the prediction accuracy significantly, we should use a design with a smaller processing time and try to keep the better prediction accuracy.

Table 8: Percentage of NERAs reduced in Model 2A

	% of NERA Reduced	
	Using Accurate Prediction	Using Inaccurate Prediction
Version I	62.71%	56.00%
Version II	65.34%	58.46%

Table 9: Percentage of outdated predictions in Model 2A

	% of Outdated Predictions	
	Using Accurate Prediction	Using Inaccurate Prediction
Version I	0.02%	43.61%
Version II	0.00%	43.63%

Table 10: Percentage of execution time reduced

	% of Execution Time Reduced	
	Using Accurate Prediction	Using Inaccurate Prediction
Version I	37.48%	28.77%
Version II	40.62%	33.35%

6 CONCLUSIONS AND FUTURE WORK

In this paper, we describe our work on interoperating AutoMod using the HLA with a new time management technique called Predictive-Conservative synchronization. The CSP interface developed for AutoMod ensures AutoMod models can be interoperated with other CSPs within the HLA framework. The new time management technique aims to improve the interoperability and efficiency of running AutoMod simulations using the HLA. It makes use of both static and dynamic information in the model to predict the happening time of external events and reduce the number of requests to advance time. The departure port mechanism helps identify the entity outgoing processes and obtains information from the model. The three phases in this approach are able to handle inaccurate predictions and choose a safe and improved value to invoke the time advancing service. A series of experiments using AutoMod models in the manufacturing domain show that the new time management approach improves the performance of the simulation significantly. This mechanism reduces the number of requests to advance time and thereby the execution time by a large percentage. The synchronization overhead can be reduced significantly by adopting this approach.

Further work includes applying the new time management approach to other CSPs, and other application domains, but the suitability and the prediction issues need to be further addressed.

ACKNOWLEDGMENT

The work was carried out as a Final Year Project in the School of Computer Engineering at Nanyang Technological University, Singapore.

REFERENCES

Brooks Automation. 2003. AutoMod User's Guide Volumes 1 and 2.

CSPI-PDG. Available via <http://www.sisostds.org> [accessed March 21, 2008].

Dahmann J. S., F. Kuhl, and R. Weatherly. 1998. Standards for Simulation: As Simple As possible but Not Simpler the High Level Architecture for Simulation, *Simulation*, 71(6):378-387.

Fujimoto, R. M. 1998. Time Management in the High Level Architecture, *Simulation*, 71(6):388-400.

Gan B. P., M. Y. H. Low, S. J. Turner, X. Wang, and S. J. E. Taylor. 2005a. Interoperating AutoSched AP Using the High Level Architecture. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 394-401. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Gan, B. P., and S. J. Turner. 2006. Interoperating Simulations Of Automatic Material Handling Systems and Manufacturing Processes. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L.F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R.M Fujimoto. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Gan, B. P., M. Y. H. Low, X. G. Wang and S. J. Turner. 2005b. Using Manufacturing Process Flow for Time Synchronization in HLA-Based Simulation. In *Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real Time Application*, Montreal, Qc. Canada, 148-157.

IEEE 1516. 2000. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA), New York, NY: Institute of Electrical and Electronics Engineers.

Nicol, D. M. 1993. The Cost of Conservative Synchronization in Parallel Discrete Event Simulations. *Journal of the Association for Computer Machinery*, 40(2):304-333.

Rohrer, M. W. 2000. AutoMod Tutorial. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J.A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 170-176. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wang, X. G., S. J. Turner, M. Y. H. Low and B. P. Gan. 2004. A Generic Architecture for The Integration of COTS Packages with the HLA, UK Operational Research Society Simulation Workshop, Birmingham, UK, 225-233.

AUTHOR BIOGRAPHIES

YUAN XI LIANG is currently a Project Officer in the Parallel and Distributed Computing Centre at Nanyang Technological University (NTU) of Singapore. He received a Bachelor of Engineering in Computer Engineering with First-Class Honors from NTU in 2008. His current research interests include parallel and distributed simulation, artificial intelligence, and image analysis and recognition. His email address is <YXLiang@ntu.edu.sg>.

STEPHEN JOHN TURNER joined Nanyang Technological University (Singapore) in 1999 and is currently a Professor in the School of Computer Engineering and Director of the Parallel and Distributed Computing Centre. Previously, he was a Senior Lecturer in Computer Science at Exeter University (UK). He received his MA in Mathematics and Computer Science from Cambridge University (UK) and his MSc and PhD in Computer Science from Manchester University (UK). His current research interests include: parallel and distributed simulation, distributed virtual environments, grid computing and multiagent systems. His e-mail address is <assjturner@ntu.edu.sg>.

BOON PING GAN is the CTO and Co-Founder of D-SIMLAB Technologies Pte Ltd. Before that he was a Senior Research Engineer with the Planning and Operations Group at the Singapore Institute of Manufacturing Technology. The focus of his research is on the application of distributed simulation technology for supply chain simulation. He received a Bachelor of Applied Science (Hons) in Computer Engineering and a Master of Applied Science from Nanyang Technological University of Singapore in 1995 and 1998, respectively. His research interests are parallel and distributed simulation, parallel programs scheduling, and application of genetic algorithms. His email address is <boonping@d-simlab.com>.